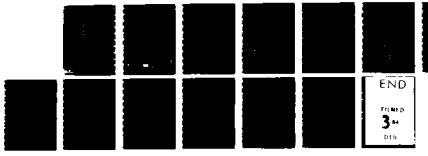
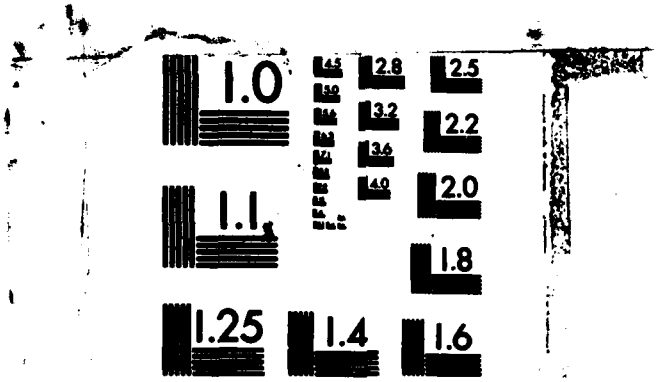
AD-A137 611 CCOMPOSITIONS FOR PERFECT GRAPHS(U) CARNEGIE-MELLON UNIV 1/1
PITTSBURGH PA MANAGEMENT SCIENCES RESEARCH GROUP
G CORNUEJOLS ET AL. OCT 83 MSRR-498 N00014-82-K-0329
UNCLASSIFIED F/G 12/1 NL





COMPOSITIONS FOR PERFECT GRAPHS

by

G. Cornuejols*
Carnegie-Mellon University

and

W. H. Cunningham**
Carleton University

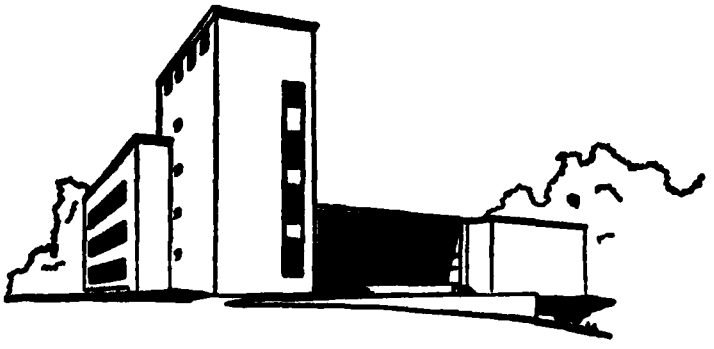
October 1983

Carnegie-Mellon University

PITTSBURGH, PENNSYLVANIA 15213

GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION

WILLIAM LARIMER MELLON, FOUNDER



Management Science Research Report No. MSRR 498

COMPOSITIONS FOR PERFECT GRAPHS

by

G. Cornuejols*
Carnegie-Mellon University

and

W. H. Cunningham**
Carleton University

October 1983

* Supported in part by NSF grant ECS-8205425 and an Alexander Von Humbolt Fellowship, while at the Institute fur Operations Research, Universitat Bonn.

** Support in part by SFB21(DFG), Institut fur Operations Research, Universitat Bonn, and by NSERC of Canada

This report was prepared as part of the activities of the Management Sciences Research Group, Carnegie-Mellon University, under Contract No. N00014-82-K-0329 NR 047-607 with the U. S. Office of Naval Research. Reproduction in whole or in part is permitted for any purpose of the U.S. Government.

Management Sciences Research Group
Graduation School of Industrial Administration
Carnegie-Mellon University
Pittsburgh, PA 15213

ABSTRACT

It is not known whether perfect graphs can be recognized in polynomial time. One attempt is to use some graph decomposition to decompose a given graph into irreducible components, i.e., components which cannot be decomposed. Perfect graphs can be recognized in polynomial time if: (1) the composition (reverse operation of the decomposition) preserves perfection; (2) reducible graphs can be decomposed in polynomial time into two smaller graphs one of which is irreducible; and (3) irreducible perfect graphs can be recognized in polynomial time. In this paper we introduce a new composition of graphs for which (1) and (2) hold. This composition generalizes clique identification, the join and the amalgam operations and, together with complementation, it encompasses all the operations preserving perfection known to us.

1. Introduction

An operation which, given two graphs G_1 and G_2, constructs a third graph G will be called a composition. We write $G = G_1 * G_2$. Conversely, a given graph G can be *-decomposed if there exist graphs G_1 and G_2 such that $G = G_1 * G_2$ and each of G_1 and G_2 has fewer nodes than G.

Perfect graphs were introduced by Berge [1] as those graphs for which, in every node induced subgraph, the size of a largest clique is equal to the chromatic number.

In a very nice paper Burlet and Fonlupt [3] defined a composition of graphs, called amalgam, and showed how to use it to characterize in polynomial time a class of perfect graphs known as Meyniel graphs [9]. Their main results are

(i) The amalgan of two Meyniel graphs is a Meyniel graph.

(ii) Conversely any Meyniel graph can be amalgam decomposed in polynomial time into "basic" Meyniel graphs.

(iii)"Basic" Meyniel graphs can be recognized in polynomial time.

It is natural to try a similar approach for the class of perfect graphs since, at present, there is no polynomial algorithm to recognize perfect graphs.

Several compositions of graphs are known to preserve perfection: union, clique identification, graph substitution [8], join ([2], [5], [7]), amalgam [3]. In this paper we describe a new composition of graphs, called the 2-amalgam, which generalizes and unifies all these compositions. In fact this operation, together with complementation, encompasses all the operations previously known to preserve perfection.

We also give a polynomial algorithm to 2-amalgam decompose a general graph or show that no such decomposition exists. For a graph with n nodes and m edges the complexity of the algorithm is $O(m^2n^2)$. To find an

amalgam decomposition, the complexity reduces to $O(mn^2)$. Algorithms of complexity $O(n^3)$ have already appeared for finding a clique cutset or a join decomposition in a graph, see [10] and [6] respectively.

2. A Graph Composition Which Preserves Perfection

Given a node v in a graph, $\Gamma(v)$ denotes the neighbor set of v, i.e. the set of nodes adjacent to v.

Given the graphs G_1 and G_2, we define the composition ϕ_{ik} as follows:

For j=1,2 let K_j be a clique of size k in G_j and, if $i \geq 1$, consider another clique with i nodes $v_1^j,\ldots,v_i^j$ in G_j disjoint from K_j such that

(i) $K_j \subseteq \Gamma(v_h^j)$ for all h=1,...,i and

(ii) $\forall\, v^j \in K_j$, $v_h^j \cup \Gamma(v_h^j) \subseteq v^j \cup \Gamma(v^j)$ for all h=1,...,i.

The composed graph $G = G_1 * G_2$ is obtained by identifying the cliques K_1 and K_2, and for each h=1,...,i, by deleting v_h^1 and v_h^2 and joining every node of $\Gamma(v_h^1)$ to every node of $\Gamma(v_h^2)$.

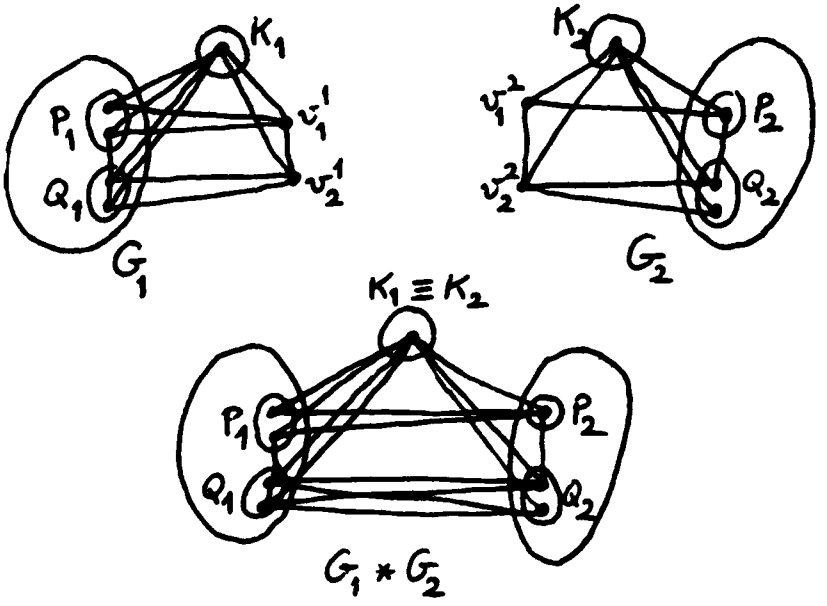


Figure 1. An example of the 2-amalgam composition.

ϕ_{00} is the union of G_1 and G_2; ϕ_{0k} is a clique identification; ϕ_{10} is the join of G_1 and G_2; ϕ_{1k} is the amalgam. ϕ_{2k} is called the <u>2-amalgam</u> of G_1 and G_2 if the following condition is satisfied (see Figure 1)

(iii) $\quad \Gamma(v_1^j) \cap \Gamma(v_2^j) = K_j$ for $j=1,2$.

The <u>2-join</u> is the special case of the 2-amalgam where $k=0$.

We shall prove that the 2-amalgam preserves perfection. We need the following lemma.

For $j=1,2$, let V_j be the node set of the graph G_j and let $U_j = V_j \setminus \{v_1^j, v_2^j\}$. Let w_j be the size of the largest clique in $G[U_j]$ (the graph induced by the node set U_j), p_j the size of the largest clique in $G[P_j]$ where $P_j \equiv \Gamma(v_1^j) \cap (U_j \setminus K_j)$ and q_j the size of the largest clique in $G[Q_j]$ where $Q_j \equiv \Gamma(v_2^j) \cap (U_j \setminus K_j)$.

Given a colouring of the nodes of a graph, $C(S)$ denotes the set of colours appearing on node set S.

<u>Lemma 1</u> If G_j is a perfect graph, then there exists a colouring of $G[U_j]$ such that $|C(U_j)| = w_j$, $|C(P_j)| = p_j$ and $|C(P_j) \cap C(Q_j)| = \max(0, p_j+q_j+k-w_k)$.

<u>Note</u>: A corollary of this lemma is that, when $p_j + q_j + k \geq w_j$, then the colouring C also satisfies $|C(Q_j)| = q_j$. (Assume not. Then p_j+k different colours appear in $P_j \cup K_j$ and more than $q_j-(p_j+q_j+k-w_j) = w_j-p_j-k$ new ones appear in Q_j, a contradiction to the fact that only w_j colours are used altogether). Therefore when $p_j+q_j+k \geq w_j$ all the colours appear in $P_j \cup Q_j \cup K_j$.

Proof of Lemma 1: Duplicate $w_j-(p_j+k)$ times the node v_1^j and $\min(w_j-(q_j+k), p_j)$ times the node v_2^j. (Duplicating zero times a node means deleting that node.) Duplication preserves perfection so the new graph, say H, is still perfect. Note that the size of its largest clique is w_j. Consider a minimum colouring of H. The conditions $C(U_j)=w_j$ and $C(P_j)=p_j$ are obviously satisfied.

If $p_j+q_j+k\leq w_j$ the duplicates of v_1^j belong to two cliques of size w_j, one with nodes from P_j and another with the duplicates of v_2^j. This shows that the p_j colours which appear in P_j must also appear on the duplicates of v_2^j. As a consequence Q_j can only be coloured with other colours, proving $|C(P_j)\cap C(Q_j)|=0$.

If $p_j+q_j+k\geq w_j$, then the size of the clique formed by the duplicates of v_1^j and v_2^j and K_j is $2w_j-(p_j+q_j+k)$ and therefore $p_j+q_j+k-w_j$ colours of C do not appear in it. Thus these colours must appear both in P_j and Q_j. This completes the proof of the lemma.

Theorem 1 The 2-amalgam preserves perfection.

Proof: Assume that G_1 and G_2 are perfect and let $G = G_1 * G_2$. The size of the largest clique in G is $\max(w_1, w_2, p_1+p_2+k, q_1+q_2+k)$. We will construct a colouring of G with the same cardinality, using colourings of G_1 and G_2 which satisfy the conditions of Lemma 1. This will be sufficient to prove that G is perfect since any node-induced subgraph of G is obtained as the 2-amalgam of the corresponding node-induced subgraphs of G_1 and G_2.

First identify the colours of $C(K_1)$ with the colours of $C(K_2)$. Let $D_j=C(P_j) \cup C(Q_j)$ and $d_j=|D_j|$ for j=1,2. Assume without loss of generality that $p_1+p_2\geq q_1+q_2$. Then either $p_1-d_1\geq q_2-d_2$ or $p_2-d_2\geq q_1-d_1$. Assume without

loss of generality that $p_1-d_1 \geq q_2-d_2$. Identify the colours of $C(Q_2) \setminus D_2$ with colours of $C(P_1) \setminus D_1$. Furthermore use the remaining colours of $C(P_1) \setminus D_1$ to colour the nodes of Q_2 which are presently coloured with colours of D_2. (This is the only step of our colouring algorithm where we actually perform a colour change. In all the other steps we perform colour identification between colours of G_1 and colours of G_2).

Note that when $p_1-d_1 \geq q_2$, then all the nodes of Q_2 are coloured with colours of $C(P_1) \setminus D_1$. Now identify the colours of $C(Q_1) \setminus D_1$ with colours of $C(P_2) \setminus D_2$ and those colours of D_2 which do not appear anymore in Q_2. Note that when $p_1-d_1 \geq q_2$ then $q_1-d_1 \leq p_2$ (as a consequence of the assumption $p_1+p_2 \geq q_1+q_2$) and therefore all the colours of $C(Q_1) \setminus D_1$ can be identified.

Continue identifying (i) the colours of $C(P_1) \cup C(Q_1)$ which are not yet identified with colours of $R_2 \equiv C(U_2) \setminus [C(P_2) \cup C(Q_2) \cup C(K_2)]$ until one of these colour sets is exhausted, (ii) the colours of $C(P_2) \cup C(Q_2)$ not yet identified with colours of $R_1 \equiv C(U_1) \setminus [C(P_1) \cup C(Q_1) \cup C(K_1)]$ until one set is exhausted, and (iii) colours of R_1 not yet identified with colours of R_2 not yet identified until one set is exhausted.

Note that we end up with a proper colouring. This colouring has been constructed so that either Q_2 is coloured only with colours of $C(P_1) \setminus D_1$ or Q_1 is coloured only with colours of D_1 and $C(P_2)$ which do not appear in Q_2. Four cases may occur.

Case 1: R_1 and R_2 are exhausted first in (i) and (ii). Then only colours of $C(P_1) \cup C(P_2) \cup C(K_1)$ are used to colour G. Namely there is a colouring of size p_1+p_2+k.

Case 2: R_2 is exhausted first in (i) and $C(P_2) \cup C(Q_2)$ is exhausted first in (ii). Then only w_1 colours are used to colour G.

Case 3: $C(P_1) \cup C(Q_1)$ is exhausted first in (i) and R_1 is exhausted first

in (ii). If the colouring in Q_2 was not modified then only w_2 colours are used to colour G. Now assume that the colouring in Q_2 was modified. This means that $d_2>0$. Therefore, as it was noted after the statement of Lemma 1, all the colours of $C(U_2)$ appear in the set $P_2 \cup Q_2 \cup K_2$. Thus only colours of $C(P_1) \cup C(P_2) \cup C(K_2)$ are used to colour G, namely p_1+p_2+k colours.

Case 4 $C(P_1) \cup C(Q_1)$ is exhausted first in (i) and $C(P_2) \cup C(Q_2)$ is exhausted first in (ii). Then depending on whether R_2 or R_1 runs out first in (iii) we are back in case 2 or case 3.

So in all cases the maximum clique of G has a cardinality equal to the colouring number of G. This completes the proof.

Note that Theorem 1 does not generalize to i-amalgams for $i\geq 3$. For example Figure 2 shows that the 3-join of two perfect graphs can contain a 7-hole.

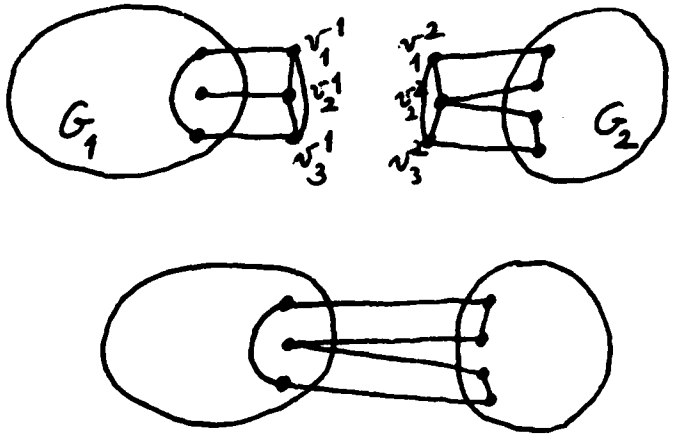


Figure 2. The 3-join composition does not preserve perfection

3. Decomposition Algorithms

Burlet and Fonlupt [3] presented an efficient algorithm either to show that a given graph G is an amalgam of smaller graphs or to show that G is not a Meyniel graph. Here we describe the first efficient algorithm to determine whether an arbitrary graph is an amalgam of smaller graphs. We also show how similar ideas can be applied to the recognition of i-amalgams, $i \geq 2$.

First, we mention previous work on algorithms for some of the more special compositions. A number of algorithms have been proposed for recognizing substitution-decomposability; the first polynomial-time one seems to be in [4]. Finding a "clique cutset", if one exists, is equivalent to determining whether a graph arises from smaller graphs by clique identification. There is an elegant and efficient algorithm for this problem [10]. Finally, a polynomial-time algorithm for recognizing join-decomposability (which includes by a simple construction recognition of substitution-decomposability) was given in [6].

In this section we let V, E denote the vertex-set and edge-set of G, and we put $n=|V|$, $m=|E|$. We assume for convenience that G is connected. Given a partition (A_1, C, A_2) of V into three sets, let B_1 denote $\{u \in A_1:$ $uv \in E$ for some $v \in A_2\}$, and similarly for B_2.

We say that (A_1, C, A_2) is an (amalgam) split of G if:

(i) $|A_1| \geq 2 \leq |A_2|$;

(ii) $uv \in E$ whenever $u,v \in C$, $u \neq v$;

(iii) $uv \in E$ whenever $u \in C$, $v \in B_1 \cup B_2$;

(iv) $uv \in E$ whenever $u \in B_1$, $v \in B_2$.

It is easy to see that G is amalgam decomposable if and only if it admits a split (A_1, C, A_2) as above. If (A_1, C, A_2) has the property

that one (and thus both) of B_1, B_2 is empty, then C is a clique cutset. We may suppose that the $O(nm)$ algorithm [10] for finding clique cutsets has already been applied, so we restrict attention here to the existence of splits (A_1, C, A_2) for which B_1 and B_2 are non-empty.

The algorithm for finding a split of G, or determining that there is none, uses ideas introduced in [6] for the case $C=\emptyset$. We give an $O(n^2)$ algorithm to determine for a fixed edge $xy \in E$, whether there is a split (A_1, C, A_2) for which $x \in A_1$, $y \in A_2$. Such an algorithm can be used to provide and $O(n^2m)$ algorithm to decide whether G has a split. (In the case $C\neq\emptyset$, the resulting algorithm is $O(n^3)$, because it is enough to run the basic algorithm for each edge xy of some spanning tree of G.)

Henceforth, we assume that $|V|\geq 4$, and we deal with a fixed edge xy of G. A preliminary step is to find a vertex $z\neq x, y$ such that no split (A_1, C, A_2) with $x \in A_1$, $y \in A_2$ satisfies $z \in C$. There is a simple procedure to find such a vertex, if G is not complete. (Of course, if G is complete, then (A_1, ϕ, A_2) is a split whenever $|A_1|\geq 2\leq |A_2|$.) Choose two non-adjacent vertices u, v. If either is not a common neighbour of x and y, then it is certainly an acceptable choice for z. In the alternative case, either of u, v may be chosen to be z. (If not, we would have $u, v \in C$ or $u \in B_1 \cup B_2$ and $v \in C$, or $u \in C$ and $v \in B_1 \cup B_2$; each of these implies that $uv \in E$.) Now any split (A_1, C, A_2) for which $x \in A_1$ and $y \in A_2$ satisfies $z \in A_1$ or $z \in A_2$, so it will be enough to give an $O(n^2)$ algorithm to solve the following problem. (It will be necessary to use that algorithm twice, once with the roles of x and y interchanged.)

(1) Problem. Find a split (A_1, C, A_2) satisfying $z, x \in A_1$, $y \in A_2$, or determine that there is none.

Consider a partition (S, K, T) of V having the following property:

(2) $x, z \in S$, $y \in T$, and $xv, yv \in E$ for all $v \in K$; moreover, for any split (A_1, C, A_2) with $x, z \in A_1$ and $y \in A_2$, we have $S \subseteq A_1$ and $A_2 \subseteq T$.

Initially, putting $S=\{x,z\}$ and $K=\phi$ determines a partition satisfying (2). On the other hand, if (S, K, T) satisfies (2) with $T=\{y\}$, then we know that there is no positive solution to (1). The algorithm maintains (S, K, T) satisfying (2) and, at each step, either recognizes that (S, K, T) is the desired split or finds an element which can be moved from T to S, from T to K, or from K to S. The rules for moving elements of V are simple, and we describe and justify them now. Henceforth, (S, K, T) always denotes a partition of V, so specifying two of these sets determines the third. Throughout, it is assumed that (S, K, T), (A_1, C, A_2) are as in (2).

Rule 1. If $u \in S$, $v \in T$, $uy \in E$, $xv \in E$, $uv \notin E$ then v can be added to S.

Justification. Since $uy \in E$, $u \in S$ we have $u \in B_1$. If $v \in C$, then $uv \in E$, a contradiction. If $v \in A_2$ then, since $xv \in E$, $v \in B_2$ and so $uv \in E$, a contradiction. Hence $v \in A_1$, as required.

Rule 2. If $u \in S$, $v \in T$, $uv \in E$, $xv \notin E$, then v can be added to S.

Justification. Clearly $x \in B_1$ and, if $v \in A_2$, then $v \in B_2$. Thus $v \in C$ or $v \in A_2$ would imply $xv \in E$, a contradiction, so $v \in A_1$.

Rule 3. If $u \in S$, $v \in T$, $uv \in E$, $uy \notin E$, then v can be added to K if $xv, yv \in E$, and otherwise v can be added to S.

Justification. Since $uy \notin E$, $u \in S$, we must have $u \in A_1 \setminus B_1$. Therefore, since $uv \in E$, we must have $v \in A_1 \cup C$. However, $v \in C$ implies $vx, vy \in E$, so if one of these fails v can be added to S, and otherwise v can be added to K.

Rule 4. If $u \in S$, $v \in K$, $uy \in E$, $uv \notin E$, then v can be added to S.

Justification. Since $uy \in E$, $u \in S$, we must have $u \in B_1$, so $v \in C$ would imply $uv \in E$.

Rule 5. If $u \in K$, $v \in T$, $xv \in E$, $uv \notin E$, then v can be added to S.

Justification. Since $u \in K$, we have $ux \in E$, so $u \in B_1 \cup C$. Since $xv \in K$, $v \in A_1 \cup C \cup B_2$, but $v \in C$ or $v \in B_2$ would imply $uv \in E$, a contradiction.

Rule 6. If $u, v \in K$, $u \neq v$ and $uv \notin E$, then u and v can be added to S.

Justification. Since $u, v \in K$, we have $uy, vy \in E$, so $u, v \in B_1 \cup C$. But if one or both of u, v are in C, then $uv \in E$, a contradiction.

Proposition. Suppose, beginning with $S=\{x,z\}$ and $K=\phi$, Rules 1 through 6 are used repeatedly until no further application is possible. If $|T| \geq 2$, then (S, K, T) is the split required in (1), and otherwise no such split exists.

Proof. The second part of the claim, that $|T|<2$ implies that no such split exists, is immediate from the facts that the initial choice of (S, K, T) satisfies (2) and that Rules 1-6 preserve (2). Now suppose that $|T| \geq 2$. We must show that $A_1=S$, $C=K$, $A_2=T$ satisfies (i)-(iv). Of course,

(i) is satisfied, and (ii) follows from the fact that Rule 6 cannot be applied. Now suppose that $u \in C$ and $v \in B_1 \cup B_2$. Since u can enter K only by Rule 3, we have $ux, uy \in E$. Now if $v \in B_1$, then since Rule 4 cannot be applied, we have $uv \in E$. Similarly, if $v \in B_2$, then since Rule 5 cannot be applied we have $uv \in E$. Therefore, (iii) is satisfied. Finally, suppose that $u \in B_1$, $v \in B_2$. By the definition of B_1, B_2 there exist $p \in A_1$, $q \in A_2$ with $uq, pv \in E$. Since Rule 2 cannot be applied, we have $xv \in E$ and, since Rule 3 cannot be applied, we have $uy \in E$. Then, since Rule 1 cannot be applied, we have $uv \in E$. Thus (iii) is proved, so (S, K, T) is a split.

It is now clear that our suggested algorithm is correct and that it will run in polynomial time. However, we claim that it can be implemented to run in time $O(n^2)$ for each choice of x, y. The preliminary step which finds z is clearly $O(n^2)$. All of Rules 1 to 6 are stated in terms of (some or all of) vertices u, v, x, y. Given the adjacency lists for each of these vertices in characteristic vector form and (S, K, T) represented by a (0, 1, -1)-vector, we can decide whether one of Rules 1 to 6 can be applied, and make any necessary change to (S, K, T) in constant time. To enable the algorithm to perform correctly with only $O(n^2)$ such operations, we process the vertices in a special order. Suppose that $u \in S$, and we want to check for applications of Rules 1 to 4. Any $v \notin S$ which cannot be added to S as a result of such an application, cannot later be added to S, using the current u. That is, we can check for all such applications, for a fixed u, at one time.

We maintain a list L_1 of elements of S to be scanned, and a list L_2 of elements of K to be scanned. Initially, $L_1=\{x,z\}$, and $L_2=\phi$. Each time an element is added to S it is added to L_1, and each time an element is

added to K it is added to L_2. When an element is scanned it is deleted from its list. Scanning an element of L_1 means asking it to play the role of u in Rules 1 to 4. Scanning an element of K means asking it to play the role of u in Rules 5 and 6. The algorithm terminates when L_1 and L_2 are empty. Clearly, every vertex is scanned at most twice, and each scanning operation requires $O(n)$ time, so we obtain the desired $O(n^2)$ bound. Since we must run this algorithm for every choice of x, y, we have an $O(n^2m)$ algorithm to find an amalgam split.

Now we consider the recognition of 2-amalgam decomposability. In this case we require that the partition (A_1, C, A_2) satisfy (ii), (iii), and (i'), (iv') below.

(i') $|A_1| \geq 3 \leq |A_2|$;

(iv') There exists a partition $\{B_{i1}, B_{i2}\}$ of B_i, i=1 and 2, such that if $u \in B_{1j}$, $v \in B_{2j}$ then $uv \in E$ if and only if j=k.

The method for finding, if possible, such a partition is a natural extension of that used for the amalgam. (As usual, we assume first that G is not decomposable with respect to any of the simpler decompositions.) Where x_1y_1, $x_2y_2 \in E$ and x_1y_2, $x_2y_1 \notin E$, we try to find (A_1, C, A_2) as above for which $x_j \in B_{1j}$, $y_j \in B_{2j}$, j=1 and 2. (Necessarily, x_1, y_1, x_2, y_2 must be distinct.) Again, it is necessary to find a vertex $z \neq x_1, y_1, x_2, y_2$ such that $z \notin C$ for any such partition. Any vertex which is not a common neighbour of x_1, y_1, x_2, y_2 will do, as will any vertex which is not adjacent to some common neighbour. If no such z exists, G has at most 5 vertices, because otherwise $\{x_1, y_1, x_2, y_2\}$ and its complement yield a join decomposition, and so G is not 2-amalgam decomposable. Any partition (A_1, C, A_2) of the kind required must satisfy either $z \in A_1$ or $z \in A_2$, so it will be enough to describe an algorithm to find (A_1, C, A_2) such that $x_1, x_2, z \in A_1$ and $y_1, y_2 \in A_2$.

We begin with $S=\{x_1, y_1, z\}$ and $C=\phi$, and apply a set of rules similar to those for the amalgam. Each of Rules 1 to 6 have analogues for the present situation. As examples, we give two of these analogues.

Rule 1'. If $u \in S$, $v \in T$, $uv \notin E$ and for some i, $uy_i \in E$, $x_i v \in E$, then v can be added to S.

Rule 5'. If $u \in K$, $v \in T$, $uv \notin E$ and, for some i, $x_i v \in E$, then v can be added to S.

We also need two new rules, both based on the requirement that $B_{i1} B_{i2}=\phi$ for i=1 and 2.

Rule 7'. If $v \in T$ and $x_1 v$, $x_2 v \in E$, then v can be added to K if vy_1, $vy_2 \in E$, and otherwise v can be added to S.

Rule 8'. If $u \in S$ and uy_1, $uy_2 \in E$, then stop; there can be no 2-amalgam split (A_1, C, A_2) with $S \subseteq A_1$, y_1, $y_2 \in A_2$.

So the algorithm can terminate by using Rule 8' as well as by encountering $T=\{y_1, y_2\}$. Similar implementation techniques to the ones described before, can be used to obtain an $O(n^2)$ time bound for this algorithm. Since there are $O(m^2)$ possible choices for x_1, y_1, x_2, y_2, we obtain an $O(n^2m^2)$ algorithm for the recognition of 2-amalgam decomposability. Similarly, there is an $O(n^2m^i)$ algorithm for i-amalgam decomposability.

It is interesting to remark that the algorithms presented in this paper—as well as those in [10], [6] for clique cutsets and join decomposability—either prove that no decomposition exists or find a

decomposition into two smaller grpahs one of which is irreducible. Therefore, at most n applications of these algorithms are needed to decompose a graph G into irreducible factors.

Finally, we mention that we may want to require that the two graphs being composed be isomorphic to induced subgraphs of the composition graph. (Then the perfection of the composition would imply, as well as be implied by, the perfection of the smaller graphs.) This requirement is automatically satisfied by clique-identification, the join and the amalgam compositions. For the 2-amalgam (and the 2-join) it is satisfied provided that there is at least one edge joining some vertex of B_{i1} to some vertex of B_{i2} for i=1 and 2. The question arises whether 2-amalgam decomposability with this additional requirement can be recognized efficiently. In fact, it can, and with the same efficiency as for the ordinary 2-amalgam. Namely, we can restrict the choice of x_1, y_1, x_2, y_2 to the case where x_1x_2, $y_1y_2 \in E$.

References

1. C. Berge, Färbung von Graphen, deren sämtliche bzw. deren ungerade Kreise stan sind, Wissenschafthiche Zeitung, Martin Luther Univ. Halle Wittenberg (1961), 114.

2. R.E. Bixby, A Composition for Perfect Graphs, technical report, Northwestern University (1982) to appear in Annals of Discrete Mathematics.

3. M. Burlet and J. Fonlupt, Polynomial Algorithm to Recognize a Meyniel Graph, Research Report 303, IMAG, University of Grenoble (1982), to appear in Annals of Discrete Mathematics.

4. D.D. Cowan, L.O. James and R.G. Stanton, Graph Decomposition for Undirected Graphs, Proceedings of the Third Southeastern Conference in Combinations, Utilitas Mathematica, Winnipeg, Canada (1972).

5. W.H. Cunningham, A Combinatorial Decomposition Theory, Thesis, University of Waterloo (1973).

6. W.H. Cunningham, Decomposition of Directed Graphs, SIAM Journal on Algebraic and Discrete Methods 3 (1982), 214-228.

7. W.H. Cunningham and J. Edmonds, A Combinatorial Decomposition Theory, Canadian Journal of Mathematics 32 (1980), 734-765.

8. L. Lovasz, Normal Hypergraphs and the Perfect Graph Conjecture, Discrete Mathematics 2 (1972), 253-267.

9. H. Meyniel, On the Perfect Graph Conjecture, Discrete Mathematics 16 (1976), 339-342.

10. S. Whitesides, An Algorithm for Finding Clique Cut-Sets, Information Processing Letters 12 (1981), 31-32.


FILM
3-8